CS583A: Course Project


```
Xiaochi Ma, Handa Shi -- May 5, 2019
```


1. Summary

We participate an active competition named TMDB Box Office Prediction.

The final model we choose is `XGBoost`. Include several numeric and date values.

Also we joined all text features like

> overview, plot keywords ,languages, production companies, and countries

into to one line text as a **text** feature for a movie. Then we use GloVe word embeddings to trans this **text** feature into a vector. At last, we use `Keras` to pretrain this vector, encode into 50 dimensions vector.

We implement the convolutional neural network using `Keras` and run the code on a Kaggle kernel.

Submissions are evaluated on Root-Mean-Squared-Logarithmic-Error (RMSLE) between the predicted value and the actual revenue. Logs are taken to not overweight blockbuster revenue movies.

In the public leaderboard, our score is top 30%. The result on the public leaderboard is not available until 05/30/2019.

2. Problem Description

Problem

The problem is to predict the revenue of movies.

This is a regression problem.

The competition is at TMDB Box Office Prediction

In this competition,we can utilize the metadata on over 7,000 past films from The Movie Database to try and predict their overall worldwide box office revenue.

Data

The data contains 3000 samples, include:

> cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries

There are numeric, date and text values.

Challenges

The training set is too small just thousands of movies.

3. Solution

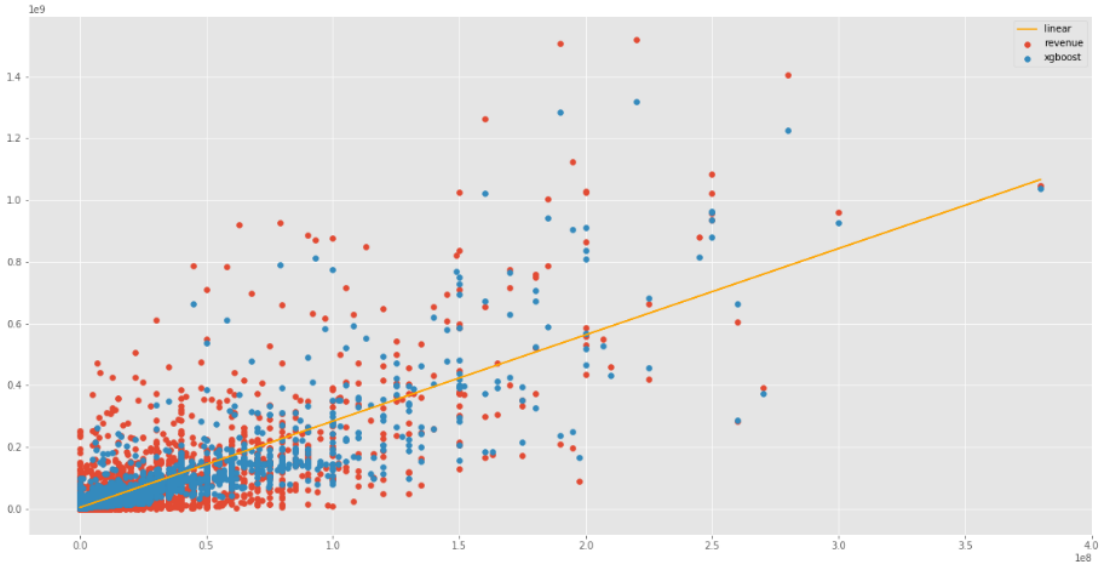
Model

The model we finally choose is the XGBRegressor and CNN, a standard deep convolutional neural network.

Implementation

1. **Baseline Model**

   We simply use linear regression with one feature - *budget* to predict the revenue:



In [18]:
```
print(np.sqrt(metrics.mean_squared_log_error(train['revenue'], li_pre)))
print(np.sqrt(metrics.mean_squared_log_error(train['revenue'], xg_pred)))
```

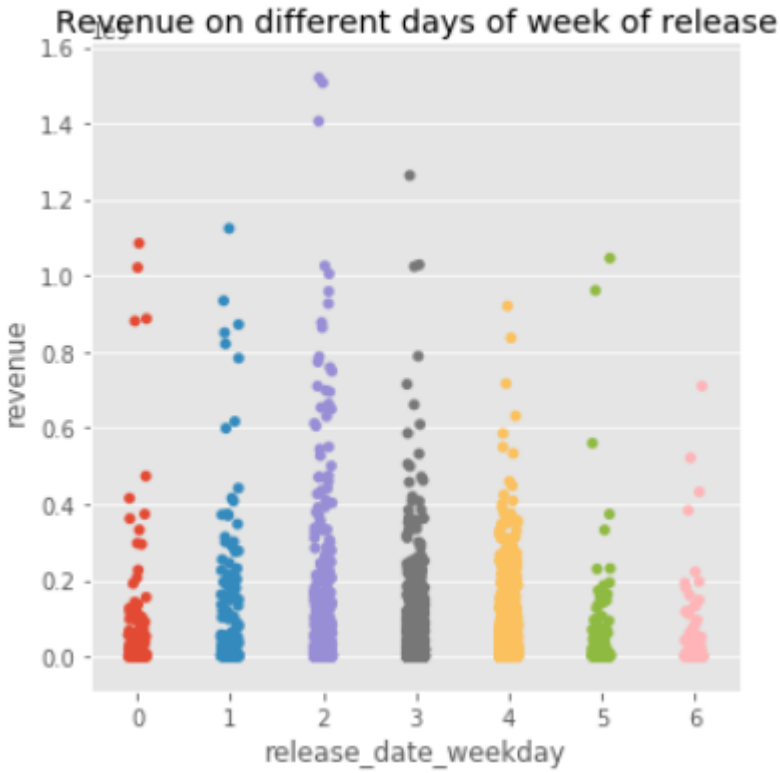
```
2.6545800150263794
2.689694204276335
```

The error is about 2.654

2. **With numeric and date values LGBoost model**

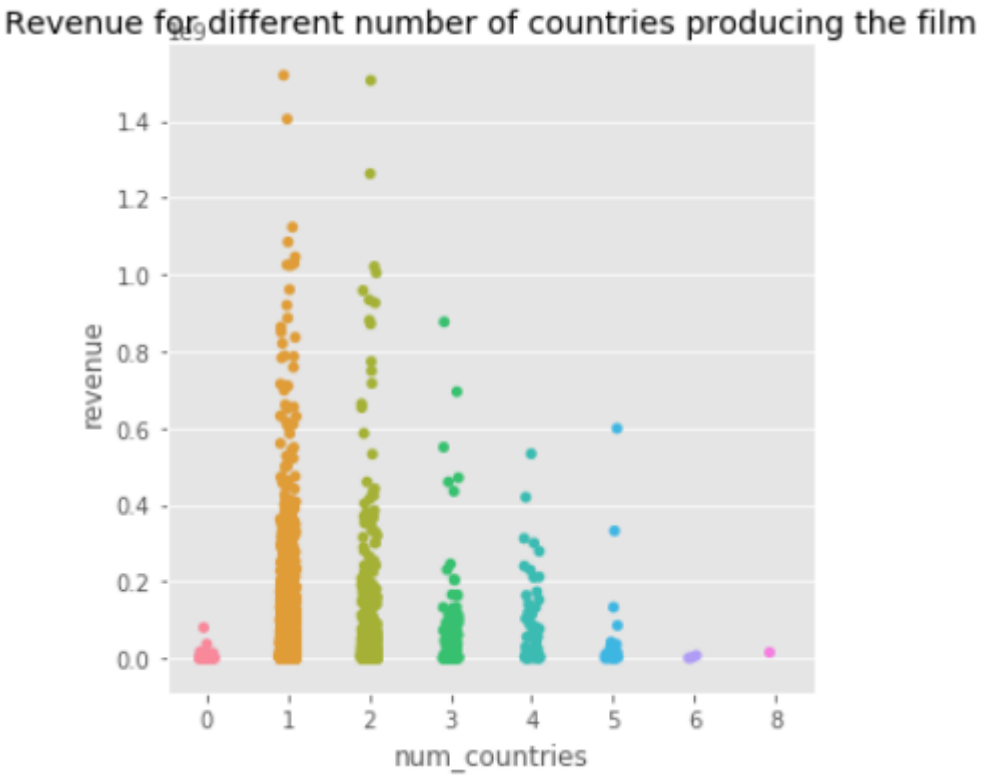
   Through feature analysis, we found some features may play more importance roles in the prediction.

   - Weekday:



If we just thing about the best record: Wednesdays and Thursdays is the best day to release your movie and Sunday is the worst day to release your movie. But if we consider the volume of the movie Wednesday also has a high volume so if we think revenue per movie the Wednesday may not be very high.

- Country:



1~2 countries is the most in the chart. So it has a higher chance to have high revenue. The more the countries are the less movies has. So it has less chance to have a high revenue.

In this round, we use **LGBoost** to make a prediction, and the error is about **2.07** which is much better than baseline.

In [49]:
```
params = {'num_leaves': 30,
          'min_data_in_leaf': 20,
          'objective': 'regression',
          'max_depth': 5,
          'learning_rate': 0.01,
          "boosting": "gbdt",
          "feature_fraction": 0.9,
          "bagging_freq": 1,
          "bagging_fraction": 0.9,
          "bagging_seed": 11,
          "metric": 'rmse',
          "lambda_l1": 0.2,
          "verbosity": -1}
model1 = lgb.LGBMRegressor(**params, n_estimators = 20000, nthread = 4, n_jobs = -1)
model1.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_valid, y_valid)], eval_metric='rmse',
        verbose=1000, early_stopping_rounds=200)
eli5.show_weights(model1, feature_filter=lambda x: x != '<BIAS>')
```

```
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[689]   training's rmse: 1.53541        valid_1's rmse: 2.07238
```

3. **Join text into vector to predict**

   We found that text features like *overview*, *keyword* may influence the revenue. So we union all text features into one line text and used CNN to encode them into a 50 dimension vector to enhance our feature matrix.

   The last error is **1.568**

In [57]:
```
train_pred = xg_model.predict(X)
train_pred = np.expm1(train_pred)
np.sqrt(mean_squared_log_error(train['revenue'], train_pred))
```

Out[57]:
```
1.5683522275612207
```

Setting

The loss function is [Root-Mean-Squared-Logarithmic-Error (RMSLE)]

The opitmizer is RMSprop.

batch_size is 128,

epochs is 20

learning_rate is 1E-3

Here is our **XGBoost**'s parameters, we used **RandomizedSearchCV** to get best accuracy.

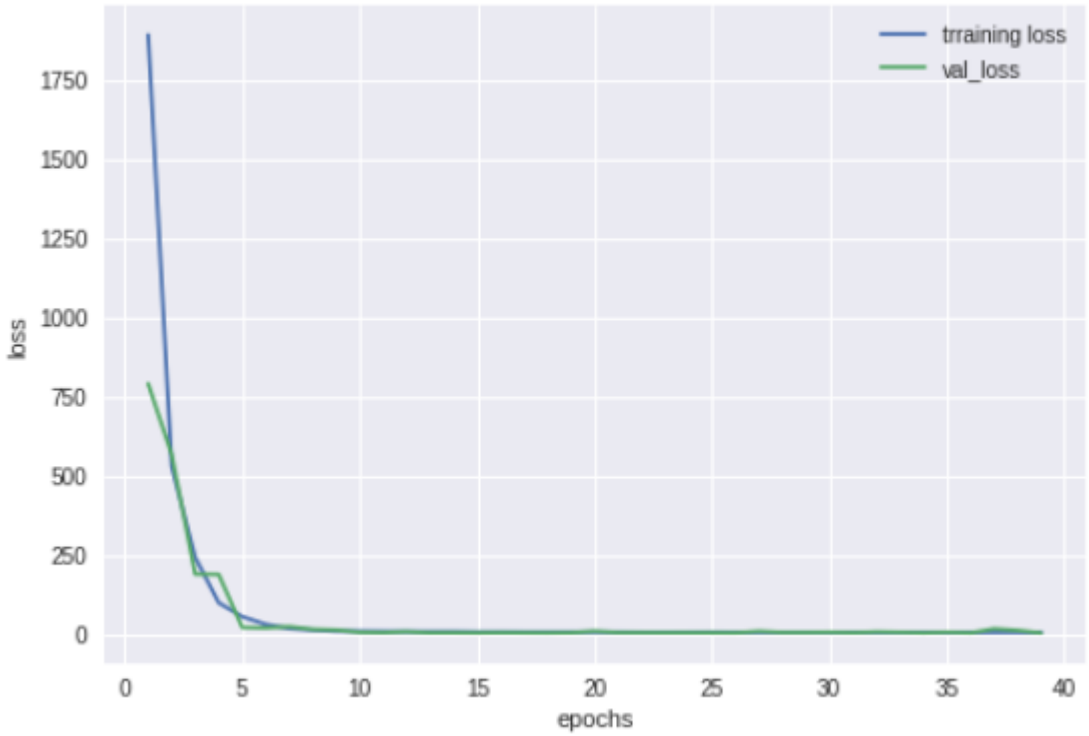
```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import StratifiedKFold
from xgboost import XGBRegressor
xg_model = XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bytree=1, gamma=4.541003990662603, importance_type='gain',
               learning_rate=0.08209238500752991, max_delta_step=0, max_depth=4,
               min_child_weight=8, missing=None, n_estimators=137, n_jobs=1,
               nthread=None, objective='reg:linear', random_state=0, reg_alpha=0,
               reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
               subsample=1)

xg_train = xg.DMatrix(data=X, label=np.array(y))
xg_model.fit(X, y)
xg_test = xg.DMatrix(data=X_test)
xg_pred = xg_model.predict(X_test)
```

Advanced tricks

We use GloVe word embeddings to trans these **text** features into a vector. At last, we use `Keras` to pretrain this vector, encode into 50 dimensions vector.

Through the pretrain, we decrease error from 2.07 to 1.57.

Cross-validation



It seems well, doesn't overfit.

4. Compared Methods

Methods	RMSLE
One feature + Linear Reg	2.65
Only with numeric and date values + LGBoost	2.07
Pretrained Text, numeric and date values + XGBoost	1.57

5. Outcome

We participated in an active competition.

We rank top 39% in the public leaderboard